

REMARKS

In the Office Action, the Examiner rejected claims 1-18 under 35 USC §112 and under 35 USC §102. These objections and rejections are fully traversed below.

Claims 1-18 have been amended to correct minor informalities and to further clarify the subject matter regarded as the invention. In addition, claims 19-28 have been added to add features not recited in the previously pending claims or further clarify features recited in the previously pending claims. Claims 1-28 are now pending.

Reconsideration of the application is respectfully requested based on the following remarks.

REJECTION OF CLAIMS 1-18 UNDER 35 USC §112

In the Office Action, the Examiner rejected claims 1-18 under 35 USC §112, first paragraph, as containing subject matter which was not described in such a way as to reasonably convey to one skilled in the relevant art that the inventor had possession of the claimed invention at the time that the application was filed. In view of the amended claims, Applicant respectfully submits that the claims as amended are supported by the subject matter in the specification. Specifically, the summary on page 4 of the specification as well as the detailed description support the claimed invention. FIG. 1 and associated text illustrate the process during which the source code (e.g., intermediate source code or “bytecodes”) and data structure identifying addresses of the portions of the source code that create local objects are generated. Specifically, intermediate source code (source code) is generated from the method (e.g., initial source code) at block 104. Generation of the data structure (optimization) is performed at block 106 (and further detailed in FIG. 2). The data structure (e.g., local table) is generated at block 204 of FIG. 2 and further illustrated in FIG. 3. FIG. 4 details one specific way to generate class files (and associated attribute_info structure of a Java class file). In other words, the attribute_info structure is one manner of implementing

the data structure claimed (e.g., of FIG. 2). FIG. 5 details one method of executing the method using information stored in a data structure such as that generated in FIG. 2 or FIG. 4. Specifically, the method is executed such that local objects and non-local objects are stored in separate heaps. While this invention is illustrated with reference to a Java frame, this example is merely illustrative. Specifically, as recited in the claims, the intermediate code generated during compile time may be referred to as source code (intermediate source code) or bytecodes. During execution, the next source code (bytecode) of the source code is obtained and interpreted (in this example). During the interpretation, the objects are stored according to whether they are classified in the data structure (e.g., storing a PC or address of the bytecodes that generate the local objects during execution of the method) as local or non-local objects. Accordingly, in view of the above, Applicant respectfully requests that the Examiner withdraw the rejection to the claims under 35 USC §112, first paragraph.

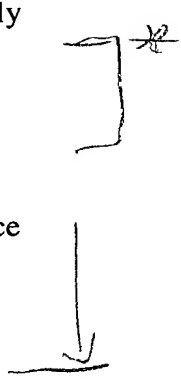
In the Office Action, the Examiner rejected claims 1-18 under 35 USC §112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which Applicant regards as the invention. Claims 1-18 are amended in accordance with the Examiner's recommendations. However, Applicant respectfully submits that "a data structure" as initially recited in the first element of claim 1 is not indefinite. Hence, Applicant respectfully requests that the Examiner withdraw the rejection of claims 1-18 under 35 USC §112, second paragraph.

REJECTION OF CLAIMS 1-18 UNDER 35 USC §102


In the Office Action, the Examiner rejected claims 1-18 under 35 USC §102(e) as being unpatentable over Pradhan et al., U.S. Patent No. 6,446,257, ('Pradhan' hereinafter).

The present invention enables local and non-local objects to be stored in separate heaps, thereby optimizing the garbage collection process. Pradhan neither discloses nor suggests the claimed invention. Pradhan does relate to garbage collection. However, Pradhan says nothing about local objects. Rather, col. 2, lines 59-60 of Pradhan indicate that the invention relates to "live object relocation." While a live object is referred to as an object that "may be globally known" in col. 1, lines 34-35, there is no reference to local and non-

local objects, or the separate storage of local and non-local objects. While the Examiner cites col. 8, lines 2-4 (source code that creates local objects) and col. 9, lines 26-30 (creating a local object on a local heap of memory), Applicant respectfully submits that col. 8, lines 2-4 merely state "A source compiler 107 processes a source code file 118 and thereby transforms the source code file 118 into an intermediate file 122." Moreover, col. 9, lines 26-30 merely state that "garbage collection works in conjunction with heap allocation..." In no manner does Pradhan disclose or suggest storing local and non-local objects in separate heaps. In fact, Pradhan discloses that a heap may be divided into two or more groups segregated by age, where the groups are referred to as generations. See Pradhan, col. 9, lines 36-40. Since segregation by age is not identical to segregation of local and non-local objects, Applicant respectfully submits that the claimed invention is not anticipated by Pradhan.

A handwritten bracket on the right side of the text, spanning from the first sentence to the last sentence of the paragraph. An arrow points downwards from the bracket.

The dependent claims depend from one of the independent claims and are therefore patentable for at least the same reasons. However, the dependent claims recite additional limitations that further distinguish them from the cited references. For instance, dependent claim 13 recites that the data structure is an attribute info structure of a Java class file. Hence, it is submitted that the dependent claims are patentably distinct from Pradhan.

A handwritten bracket on the right side of the text, spanning from the sentence about claim 13 to the concluding sentence of the paragraph.

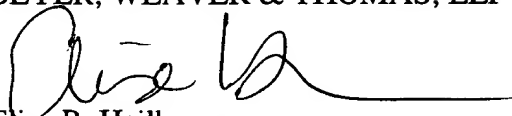
Based on the foregoing, it is submitted that the independent claims are patentably distinct from Pradhan. In addition, it is submitted that the dependent claims are also patentably distinct for at least the same reasons. The additional limitations recited in the independent claims or the dependent claims are not further discussed as the above discussed limitations are clearly sufficient to distinguish the claimed invention from Pradhan. Thus, it is respectfully requested that the Examiner withdraw the rejection of claims 1-18 under 35 USC §102(e).

Reconsideration of the application and an early Notice of Allowance are earnestly solicited. If there are any issues remaining which the Examiner believes could be resolved through either a Supplemental Response or an Examiner's Amendment, the Examiner is respectfully requested to contact the undersigned attorney at the telephone number listed below.

Applicants hereby petition for an extension of time which may be required to maintain the pendency of this case, and any required fee for such extension or any further fee required in connection with the filing of this Amendment is to be charged to Deposit Account No. 50-0388 (Order No. SUN1P287).

Respectfully submitted,

BEYER, WEAVER & THOMAS, LLP

A handwritten signature in black ink, appearing to read 'Elise R. Heilbrunn', followed by a long horizontal line extending to the right.

Elise R. Heilbrunn

Reg. No. 42,649

BEYER, WEAVER & THOMAS, LLP

P.O. Box 778

Berkeley, CA 94704-0778

Tel. (510) 843-6200

MARKED-UP COPY OF CLAIMS

1. (Once Amended) A method of executing a method to enable memory associated with objects not referenced external to the executed method to be reclaimed upon completion of execution of the executed method, comprising:

obtaining a data structure including one or more addresses of source code of the method that creates one or more local objects, the one or more local objects being created during execution of the method and are not referenced outside the method;

obtaining next source code in the source code of the method;

determining whether an address of the obtained next source code is in the data structure; [and]

when the address of the obtained next source code is in the data structure including one or more addresses of source code of the method that creates one or more local objects, creating a local object on a local heap of memory using the source code associated with the address of the obtained next source code such that local objects are stored in memory separately from non-local objects[.] ; and

when the address of the obtained next source code is not in the data structure including one or more addresses of source code of the method that creates one or more local objects, creating a non-local object on a non-local heap of memory using the source code associated with the address of the obtained next source code such that non-local objects are stored in memory separately from local objects.

2. The method as recited in claim 1, wherein determining whether the address of the source code is in the data structure comprises:

determining whether a program counter of the source code is in the data structure.

3. (Once Amended) The method as recited in claim 1, further comprising:

when the address of the obtained next source code is in the data structure, reclaiming memory associated with the local heap upon termination of execution of the method.

4. (Once Amended) The method as recited in claim 1, further comprising:

when the address of the obtained next source code is in the data structure, returning

memory associated with the local heap to a pool of available memory upon termination of execution of the method.

5. The method as recited in claim 1, further comprising:
creating a dynamic structure adapted for storing dynamic information related to method execution; and
associating the local heap with the dynamic structure.
6. The method as recited in claim 5, wherein associating the local heap with the dynamic structure comprises extending a pointer from the dynamic structure to the local heap.
7. The method as recited in claim 5, wherein the dynamic structure is a Java frame.
8. The method as recited in claim 7, wherein the Java frame is a data structure in a Java interpreter.
9. The method as recited in claim 8, further comprising:
removing the Java frame from memory when execution of the method terminates.
10. The method as recited in claim 1, further comprising:
allocating a free chunk of available memory as the local heap for storage of one or more local objects.
11. The method as recited in claim 1, wherein the local heap comprises one or more chunks of memory, wherein creating a local object on a local heap of memory using the source code comprises:
determining whether the local heap contains available memory for storage of the local object;
when the local heap contains available memory sufficient for storage of the local object, creating the local object in one of the chunks of memory;
when the local heap does not contain available memory sufficient for storage of the local object, allocating a new chunk, associating the new chunk with the local heap, and storing the local object in the new chunk.

12. The method as recited in claim 11, wherein associating the new chunk with the local heap comprises providing a pointer to the new chunk such that the local heap is composed of a linked list of memory chunks.

13. The method as recited in claim 1, wherein obtaining a data structure including one or more addresses of source code that creates local objects comprises:

obtaining an attribute_info structure from a Java class file.

14. The method as recited in claim 1, wherein the source code comprises bytecodes.

15. The method as recited in claim 8, wherein the bytecodes are Java bytecodes.

16. (Once Amended) A computer-readable medium for executing a method to enable memory associated with objects not referenced external to the executed method to be reclaimed upon completion of execution of the executed method, comprising:

instructions for obtaining a data structure including one or more addresses of source code of the method that creates one or more local objects, the one or more local objects being created during execution of the method and are not referenced outside the method;

instructions for obtaining next source code in the source code of the method;

instructions for determining whether an address of the obtained next source code is in the data structure; [and]

instructions for when the address of the obtained next source code is in the data structure including one or more addresses of source code of the method that creates one or more local objects, creating a local object on a local heap of memory using the source code associated with the address of the obtained next source code such that local objects are stored in memory separately from non-local objects[.] ; and

instructions for when the address of the obtained next source code is not in the data structure including one or more addresses of source code of the method that creates one or more local objects, creating a non-local object on a non-local heap of memory using the source code associated with the address of the obtained next source code such that non-local objects are stored in memory separately from local objects.

17. (Once Amended) An apparatus for executing a method to enable memory associated with objects not referenced external to the executed method to be reclaimed upon completion of execution of the executed method, comprising:

means for obtaining a data structure including one or more addresses of source code of the method that creates one or more local objects, the one or more local objects being created during execution of the method and are not referenced outside the method;

means for obtaining next source code in the source code of the method;

means for determining whether an address of the obtained next source code is in the data structure; [and]

means for when the address of the obtained next source code is in the data structure including one or more addresses of source code of the method that creates one or more local objects, creating a local object on a local heap of memory using the source code associated with the address of the obtained next source code such that local objects are stored in memory separately from non-local objects[.] ; and

means for when the address of the obtained next source code is not in the data structure including one or more addresses of source code of the method that creates one or more local objects, creating a non-local object on a non-local heap of memory using the source code associated with the address of the obtained next source code such that non-local objects are stored in memory separately from local objects.

18. (Once Amended) An apparatus for executing a method to enable memory associated with objects not referenced external to the executed method to be reclaimed upon completion of execution of the executed method, comprising:

a processor; and

a memory, at least one of the processor and the memory being adapted for:

obtaining a data structure including one or more addresses of source code of the method that creates one or more local objects, the one or more local objects being created during execution of the method and are not referenced outside the method;

obtaining next source code in the source code of the method;

determining whether an address of the obtained next source code is in the data structure; [and]

when the address of the obtained next source code is in the data structure including

one or more addresses of source code of the method that creates one or more local objects, creating a local object on a local heap of memory using the source code associated with the address of the obtained next source code such that local objects are stored in memory separately from non-local objects[.]; and

when the address of the obtained next source code is not in the data structure including one or more addresses of source code of the method that creates one or more local objects, creating a non-local object on a non-local heap of memory using the source code associated with the address of the obtained next source code such that non-local objects are stored in memory separately from local objects.

19. The method as recited in claim 1, further comprising:
compiling the method to generate the data structure.
20. The method as recited in claim 19, wherein the source code is generated when the method is compiled.
21. The method as recited in claim 20, wherein the source code comprises bytecodes.
22. The method as recited in claim 3, wherein reclaiming memory is performed during garbage collection.
23. The method as recited in claim 22, wherein the garbage collection is mark and sweep garbage collection.
24. The method as recited in claim 13, further comprising:
performing class file generation such that information from the data structure is stored in the attribute_info structure of the Java class file.
25. The method as recited in claim 13, further comprising:
performing class file generation such that the data structure is generated.

26. The method as recited in claim 25, wherein the data structure is an attribute_info structure of the Java class file.

27. The method as recited in claim 1, further comprising:
generating the data structure.

28. The method as recited in claim 27, further comprising:
performing live-dead analysis on the source code to identify a set of dead objects,
each of the set of dead objects being a local object.